```
/*
Display.cpp - A simple track GPS to SD card logger. Display module.
TinyTrackGPS v0.10

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "Display.h"

Display::Display(Display_Type t):_screen(t){
    //if (_screen == SDD1306_128X64){
        _width = 16;
        _height = (_screen > 0) ? 2 : 8;
        //_offset = 0;
    //} else if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
    //    _width = 16;
    //    _height = 2;
        //_offset = 0;
    //}
}

void Display::start(){
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
        #if defined(DISPLAY_TYPE_LCD_16X2)
        lcd = new LiquidCrystal(LCD_RS, LCD_ENABLE, LCD_D0, LCD_D1, LCD_D2, LCD_D3);
        lcd->begin(_width, _height);
        #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd = new LiquidCrystal_I2C(I2C,_width,_height);
        lcd->init();
        lcd->backlight();
        #endif

        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        // DEFINICION DE CARACTERES PERSONALIZADOS
        static byte alt[8] = { 0x04, 0x0E, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04 };
        static byte ant[8] = { 0x0E, 0x11, 0x15, 0x11, 0x04, 0x04, 0x0E, 0x00 };
        static byte sd[8] = { 0x0E, 0x11, 0x1F, 0x00, 0x00, 0x17, 0x15, 0x1D };
        static byte hourglass_0[8] = { 0x1F, 0x0E, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
0x1F };
        static byte hourglass_1[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
0x1F };
```

```
        static byte hourglass_2[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
        static byte hourglass_3[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
        static byte hourglass_4[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0E, 0x0E,
0x1F };
        lcd->createChar(0, hourglass_0);
        lcd->createChar(1, hourglass_1);
        lcd->createChar(2, hourglass_2);
        lcd->createChar(3, hourglass_3);
        lcd->createChar(4, hourglass_4);
        lcd->createChar(5, alt);
        lcd->createChar(6, ant);
        lcd->createChar(7, sd);
        #endif
    //}

    //if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
SDA);
        u8x8_SSD1306->begin();
        u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
        #endif
    //}
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
    display = new DisplaySSD1306_128x64_I2C(-1);
    display->begin();
    //display->setFixedFont(ssd1306xled_font8x16);
    //display->setFixedFont(ssd1306xled_font6x8);
    display->setFixedFont(TinyTrackGPS_font8x16);
    this->clr();
    #endif
}

void Display::clr(){
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->clear();
        #endif
    //}
    //else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->clear();
        #endif
    //}
        #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display->clear();
        #endif
}

void Display::print(int vertical, int horizontal, const char text[]){
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(vertical, horizontal);
        this->print(text);
        #endif
    //}
    //else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
```

```
        //u8x8_SSD1306->drawString(vertical, (horizontal*2),text);
        u8x8_SSD1306->setCursor(vertical, (horizontal*2));
        this->print(text);
        #endif
    //}
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display->setTextCursor((vertical*8),(horizontal*16));
        this->print(text);
        //display->printFixed((vertical*8),(horizontal*16),text);
    #endif
}

void Display::print(int line, const char text[]){
    byte pos = _width -(strlen(text));
    pos = (pos >> 1);
    this->print((int)pos, line, text);
}

void Display::print(const char text[]){
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
    lcd->print(text);
    #endif
    //}
    //else if (_screen == SDD1306_128X64) {
    #if defined(DISPLAY_TYPE_SDD1306_128X64)
    u8x8_SSD1306->print(text);
    u8x8_SSD1306->flush();
    #endif
    //}
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
    display->write(text);
    #endif
}

void Display::print(const char text1[], const char text2[]){
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        this->print((_screen > 0)?0:1, text1);
        this->print((_screen > 0)?1:2, text2);
    //}
    //else if (_screen == SDD1306_128X64) {
    //    this->print(1, text1);
    //    this->print(2, text2);
    //}
}

void Display::print(const char text1[], const char text2[], const char text3[]){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        this->print(text1, text2);
        delay(750);
        //for (unsigned long start = millis(); millis() - start < 750;) {}
        //unsigned long start = millis();
        //do {} while (millis() - start < 750);

        this->clr();
        this->print(0,text3);
    //}
    #endif
```

```
    #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
    //else if (_screen == SDD1306_128X64) {
        this->print(0, text1);
        this->print(1, text2);
        this->print(2, text3);
    //}
    #endif
}

void Display::print(const char text1[], const char text2[], const char text3[],
const char text4[]){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
    this->print(text1,text2);
    delay(750);
    this->print(text3,text4);
    #endif
    #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
    this->print(0, text1);
    this->print(1, text2);
    this->print(2, text3);
    this->print(3, text4);
    #endif
}

void Display::wait_anin(unsigned int t){
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(15,1);
        lcd->write((byte)t%5);
        #endif
    //}
    //else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)

        const char p[4] = {(char)47,(char)45,(char)92,(char)124};
        u8x8_SSD1306->setCursor((_width-1),6);
        u8x8_SSD1306->print(p[t%4]);

        /*
        static uint8_t hourglass_UP[5][8] = {
0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
                                        0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
                                        0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
                                        0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
                                        0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
                                        };

        static uint8_t hourglass_DOWN[5][8] =
{0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
                                        0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
                                        0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
                                        0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
                                        0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
                                        };
        u8x8_SSD1306->drawTile((_width-1), 6, 1, hourglass_UP[t%5]);
        u8x8_SSD1306->drawTile((_width-1), 7, 1, hourglass_DOWN[t%5]);
        */
        #endif
```

```
    //}
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        //const char p[5] = {(char)58,(char)59,(char)60,(char)61,(char)62};
        display->setTextCursor(120,48);
        display->printChar((char)(t%5)+58);

        //display->drawWindow(0,0,0,0,"GPS Signal",true);
        //display->drawProgressBar( t%100 );

        //display->setTextCursor((_width-1)*8,48);
        //display->printChar('-');
    #endif
}

void Display::print_PChar(byte c) {
    //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
    lcd->write(c);
    #endif
    //}
    //else if (_screen == SDD1306_128X64) {
    #if defined(DISPLAY_TYPE_SDD1306_128X64)

    static uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
                                      0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
                                      0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
                                      };
    static uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
                                      0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
                                      0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8
                                      };
    //char tile;
    if (c == 5) {
        //tile = (char)0x18;
        u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
        u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
        //u8x8_SSD1306->setCursor(9, 2);
    }
    else if (c == 6) {
        //tile = (char)0x7f;
        u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
        u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
        //u8x8_SSD1306->setCursor(11, 0);
    }
    else if (c == 7) {
        //tile = (char)0xda;
        u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
        u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
        //u8x8_SSD1306->setCursor(15, 0);
    }
    //u8x8_SSD1306->print(tile);
    #endif
    //}
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        //const char p[3] = {(char)94,(char)95,(char)96};
        /*
        switch (c) {
        case '5':
            display->setTextCursor(72,32);
            break;
```

```
        case '6':
            display->setTextCursor(88,0);
            break;
        case '7':
            display->setTextCursor(120,0);
            break;
        }
        */
        /*
        if( c == 5) {
            display->setTextCursor(72,32);
            //display->printChar(p[0]);
        }
        else if( c == 6) {
            display->setTextCursor(88,0);
            //display->printChar(p[1]);
        }
        else if( c == 7) {
            display->setTextCursor(120,0);
            //display->printChar(p[2]);
        }
        */
        //#ifndef __LGT8F__
        display->printChar((char)(c+86));
        //#endif
    #endif
}

void Display::DrawLogo() {
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
    display->drawBitmap1(0,16,128,32,Logo);
    #endif
}

#if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
const PROGMEM uint8_t TinyTrackGPS_font8x16[] = {
    0x00, // 0x00 means fixed font type - the only supported by the library
    0x08, // 0x08 = 8 - font width in pixels
    0x10, // 0x10 = 16 - font height in pixels
    0x2d,

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, // - 45
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, // . 46
    0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x00, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x00, // '/'->'=' 47
    0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20,
0x10, 0x0F, 0x00, // 0 48
    0x00, 0x10, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
0x20, 0x00, 0x00, // 1 49
    0x00, 0x70, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x30, 0x28, 0x24, 0x22,
0x21, 0x30, 0x00, // 2 50
    0x00, 0x30, 0x08, 0x88, 0x88, 0x48, 0x30, 0x00, 0x00, 0x18, 0x20, 0x20, 0x20,
0x11, 0x0E, 0x00, // 3 51
    0x00, 0x00, 0xC0, 0x20, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x07, 0x04, 0x24, 0x24,
0x3F, 0x24, 0x00, // 4 52
    0x00, 0xF8, 0x08, 0x88, 0x88, 0x08, 0x08, 0x00, 0x00, 0x19, 0x21, 0x20, 0x20,
0x11, 0x0E, 0x00, // 5 53
```

```
    0x00, 0xE0, 0x10, 0x88, 0x88, 0x18, 0x00, 0x00, 0x0F, 0x11, 0x20, 0x20, 0x11, 0x0E, 0x00, // 6 54
    0x00, 0x38, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, // 7 55
    0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21, 0x22, 0x1C, 0x00, // 8 56
    0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x31, 0x22, 0x22, 0x11, 0x0F, 0x00, // 9 57

    0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80, // Code for char num 58 :
    0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80, // Code for char num 59 ;
    0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80, // Code for char num 60 <
    0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80, // Code for char num 61 =
    0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01,0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80, // Code for char num 62 >

    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // Code for char num 63 ? ->' '
    0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x20,0x3F,0x20,0x00,0x3F,0x20,0x00,0x3F, // Code for char num 64 @ ->'m'

    0x00, 0x00, 0xC0, 0x38, 0xE0, 0x00, 0x00, 0x00, 0x20, 0x3C, 0x23, 0x02, 0x02, 0x27, 0x38, 0x20, // A 33
    0x08, 0xF8, 0x88, 0x88, 0x88, 0x70, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x11, 0x0E, 0x00, // B 34
    0xC0, 0x30, 0x08, 0x08, 0x08, 0x08, 0x38, 0x00, 0x07, 0x18, 0x20, 0x20, 0x20, 0x10, 0x08, 0x00, // C 35
    0x08, 0xF8, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x10, 0x0F, 0x00, // D 36
    0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x23, 0x20, 0x18, 0x00, // E 37
    0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03, 0x00, 0x00, 0x00, // F 38
    0xC0, 0x30, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x07, 0x18, 0x20, 0x20, 0x22, 0x1E, 0x02, 0x00, // G 39
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x21, 0x01, 0x01, 0x21, 0x3F, 0x20, // H 40
    0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00, // I 41
    0x00, 0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0xC0, 0x80, 0x80, 0x80, 0x7F, 0x00, 0x00, 0x00, // J 42
    0x08, 0xF8, 0x88, 0xC0, 0x28, 0x18, 0x08, 0x00, 0x20, 0x3F, 0x20, 0x01, 0x26, 0x38, 0x20, 0x00, // K 43
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x20, 0x30, 0x00, // L 44
    0x08, 0xF8, 0xF8, 0x00, 0xF8, 0xF8, 0x08, 0x00, 0x20, 0x3F, 0x00, 0x3F, 0x00, 0x3F, 0x20, 0x00, // M 45
    0x08, 0xF8, 0x30, 0xC0, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x20, 0x00, 0x07, 0x18, 0x3F, 0x00, // N 46
    0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x20, 0x10, 0x0F, 0x00, // O 47
    0x08, 0xF8, 0x08, 0x08, 0x08, 0x08, 0xF0, 0x00, 0x20, 0x3F, 0x21, 0x01, 0x01, 0x01, 0x00, 0x00, // P 48
    0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x18, 0x24, 0x24, 0x38, 0x50, 0x4F, 0x00, // Q 49
```

```
    0x08, 0xF8, 0x88, 0x88, 0x88, 0x88, 0x70, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03, 0x0C, 0x30, 0x20, // R 50
    0x00, 0x70, 0x88, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x38, 0x20, 0x21, 0x21, 0x22, 0x1C, 0x00, // S 51
    0x18, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x18, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x00, 0x00, // T 52
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x00, 0x1F, 0x20, 0x20, 0x20, 0x20, 0x1F, 0x00, // U 53
    0x08, 0x78, 0x88, 0x00, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x07, 0x38, 0x0E, 0x01, 0x00, 0x00, // V 54
    0xF8, 0x08, 0x00, 0xF8, 0x00, 0x08, 0xF8, 0x00, 0x03, 0x3C, 0x07, 0x00, 0x07, 0x3C, 0x03, 0x00, // W 55
    0x08, 0x18, 0x68, 0x80, 0x80, 0x68, 0x18, 0x08, 0x20, 0x30, 0x2C, 0x03, 0x03, 0x2C, 0x30, 0x20, // X 56
    0x08, 0x38, 0xC8, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x00, 0x00, // Y 57
    0x10, 0x08, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x20, 0x38, 0x26, 0x21, 0x20, 0x20, 0x18, 0x00, // Z 58


 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,// Code for char num 91 '['

 0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,// Code for char num 92 '\'

 0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78,0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8,// Code for char num 93 ']'
/*
    0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x19, 0x24, 0x22, 0x22, 0x22, 0x3F, 0x20, // a 65
    0x08, 0xF8, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x11, 0x20, 0x20, 0x11, 0x0E, 0x00, // b 66
    0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x0E, 0x11, 0x20, 0x20, 0x20, 0x11, 0x00, // c 67
    0x00, 0x00, 0x00, 0x80, 0x80, 0x88, 0xF8, 0x00, 0x00, 0x0E, 0x11, 0x20, 0x20, 0x10, 0x3F, 0x20, // d 68
    0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x1F, 0x22, 0x22, 0x22, 0x22, 0x13, 0x00, // e 69
    0x00, 0x80, 0x80, 0xF0, 0x88, 0x88, 0x88, 0x18, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00, // f 70
    0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x6B, 0x94, 0x94, 0x94, 0x93, 0x60, 0x00, // g 71
    0x08, 0xF8, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x20, 0x3F, 0x21, 0x00, 0x00, 0x20, 0x3F, 0x20, // h 72
    0x00, 0x80, 0x98, 0x98, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00, // i 73
    0x00, 0x00, 0x00, 0x80, 0x98, 0x98, 0x00, 0x00, 0x00, 0xC0, 0x80, 0x80, 0x80, 0x7F, 0x00, 0x00, // j 74
    0x08, 0xF8, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x20, 0x3F, 0x24, 0x02, 0x2D, 0x30, 0x20, 0x00, // k 75
    0x00, 0x08, 0x08, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00, // l 76
    0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x3F, 0x20, 0x00, 0x3F, // m 77
    0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x20, 0x3F, 0x21, 0x00, 0x00, 0x20, 0x3F, 0x20, // n 78
    0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x1F, 0x20, 0x20, 0x20, 0x20, 0x1F, 0x00, // o 79
```

```
398     0x80, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xA1, 0x20, 0x20,
    0x11, 0x0E, 0x00, // p 80
399     0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x0E, 0x11, 0x20, 0x20,
    0xA0, 0xFF, 0x80, // q 81
400     0x80, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x20, 0x20, 0x3F, 0x21, 0x20,
    0x00, 0x01, 0x00, // r 82
401     0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x33, 0x24, 0x24, 0x24,
    0x24, 0x19, 0x00, // s 83
402     0x00, 0x80, 0x80, 0xE0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x20,
    0x20, 0x00, 0x00, // t 84
403     0x80, 0x80, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x1F, 0x20, 0x20, 0x20,
    0x10, 0x3F, 0x20, // u 85
404     0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x01, 0x0E, 0x30, 0x08,
    0x06, 0x01, 0x00, // v 86
405     0x80, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x80, 0x0F, 0x30, 0x0C, 0x03, 0x0C,
    0x30, 0x0F, 0x00, // w 87
406     0x00, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x20, 0x31, 0x2E, 0x0E,
    0x31, 0x20, 0x00, // x 88
407     0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x81, 0x8E, 0x70, 0x18,
    0x06, 0x01, 0x00, // y 89
408     0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x21, 0x30, 0x2C, 0x22,
    0x21, 0x30, 0x00, // z 90
409 */
410 };
411
412 //------------------------------------------------------------------------------
413 // File generated by LCD Assistant
414 // http://en.radzio.dxp.pl/bitmap_converter/
415 //------------------------------------------------------------------------------
416
417 const PROGMEM uint8_t Logo[] = {
418 0x00, 0x00, 0x80, 0xC0, 0x60, 0x10, 0x98, 0x4C, 0x64, 0x26, 0x12, 0x12, 0x0B, 0x09,
    0x09, 0x49,
419 0x49, 0x09, 0x09, 0x0B, 0x12, 0x12, 0x26, 0x64, 0x48, 0x98, 0x10, 0x60, 0xC0, 0x00,
    0x00, 0x00,
420 0x00, 0x00, 0x00, 0x18, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0xE8, 0x00,
    0x00, 0x00,
421 0x00, 0x00, 0xE0, 0x20, 0x20, 0xE0, 0x00, 0x00, 0x00, 0x20, 0xE0, 0x00, 0xC0, 0x20,
    0x00, 0x00,
422 0x00, 0x00, 0x18, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x20, 0x20,
    0x00, 0x00,
423 0x00, 0x00, 0x40, 0x60, 0xA0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x20, 0x20, 0x60,
    0x00, 0x00,
424 0x00, 0xF8, 0x80, 0xC0, 0x20, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x18, 0x08, 0x08, 0x98,
    0x00, 0x00,
425 0x00, 0x00, 0xF8, 0x48, 0x48, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x68, 0x48,
    0x98, 0x00,
426 0xF0, 0x1E, 0x03, 0xF0, 0x0C, 0x03, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0,
    0x60, 0xB0,
427 0x10, 0x18, 0x08, 0xC4, 0x64, 0x1E, 0x07, 0x03, 0x00, 0x81, 0x03, 0x0C, 0xF0, 0x03,
    0x1E, 0xF0,
428 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02,
    0x00, 0x00,
429 0x00, 0x00, 0x03, 0x02, 0x00, 0x03, 0x02, 0x00, 0x00, 0x08, 0x08, 0x07, 0x00, 0x00,
    0x00, 0x00,
430 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x00,
    0x00, 0x00,
431 0x00, 0x00, 0x03, 0x02, 0x02, 0x03, 0x02, 0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x02,
    0x00, 0x00,
```

```
0x00, 0x03, 0x02, 0x01, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x02, 0x03, 0x00, 0x00,
0x00, 0x00, 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x02, 0x02, 0x03, 0x00,
0x0F, 0x78, 0xC0, 0x0F, 0x30, 0xC0, 0x81, 0x00, 0xC0, 0xE0, 0x78, 0x26, 0x23, 0x10, 0x18, 0x08,
0x0D, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81, 0xC0, 0x30, 0x0F, 0xC0, 0x78, 0x0F,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x70, 0x3C, 0x03, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3E, 0x41, 0x41, 0x63, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
0x41, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x41, 0x41, 0x63, 0x1C, 0x00, 0x00, 0x00,
0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x43, 0x61, 0x59, 0x46, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x03, 0x06, 0x08, 0x19, 0x32, 0x26, 0x64, 0x48, 0x48, 0xD0, 0x90, 0x90, 0x92,
0x92, 0x90, 0x90, 0xD0, 0x48, 0x48, 0x64, 0x26, 0x32, 0x19, 0x08, 0x06, 0x03, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

#endif
```